LICEUL TEORETIC “ONISIFOR GHIBU”

SIBIU

PROIECT PENTRU ATESTARE

PROFESIONALĂ

an școlar: 2024-2025

COMPETENȚE CERTIFICATE:

I. Realizarea design-ului și structurii produselor

soft necesare implementării de: sisteme

software, aplicații software, baze de date,

pagini WEB. (software orientat client)

II. Particularizarea, configurarea și modificarea

aplicațiilor software, în scopul adaptării

sistemelor informaționale ale clientului.

ELEV: Monoran David

CLASA: a XII-a I

PROFESOR COORDONATOR : Avram Monica

TEMA PROIECTULUI:

Sistem de prezentă pe baza cardului RFID în C# și arduino.

CUPRINS

[Motivarea alegeri temei 3](#_Toc195601058)

[Aspecte teoretice 3](#_Toc195601059)

[Software utilizat 3](#_Toc195601060)

[Prezentare software 4](#_Toc195601061)

[Prezentarea aplicației 5](#_Toc195601062)

[WebApp 5](#_Toc195601063)

[WebApi 9](#_Toc195601064)

[Apelarea funcțiilor de inserare 10](#_Toc195601065)

[Inserarea de date 11](#_Toc195601066)

[Baza de date 12](#_Toc195601067)

[Securitate 13](#_Toc195601068)

[Hardware-ul 14](#_Toc195601069)

[Concluzii 16](#_Toc195601070)

[Webografie 17](#_Toc195601071)

# Motivarea alegeri temei

Am ales ca temă pentru atestat crearea unui sistem de prezență pentru ca acesta are aplicații practice nenumărate, putând fi introdus în companii ca un mod de monitorizare a prezenței angajaților sau a sălilor accesate de aceștia, de la a fi folosit de cluburi sportive de Yoga sau arte, până la contorizarea prezenței studenților.

# Aspecte teoretice

Proiectul de atestat este format din 3 componente:

* Web API-ul a fost realizat în C# în mediul de programare „Visual Studio 2022” și accesează un server de baze de date SQL pentru a stoca informații precum detaliile utilizatorilor, prezența acestora și altele.
* Web App-ul a fost realizat cu Razor Pages, care îmbină structura paginii scrisă în HTML și script-urile aferente scrise în C#.
* Hardware-ul, care constă dintr-un LCD, cititor RFID și un microcontroller ESP32 programat în C++ în ArduinoIDE.

# Software utilizat

* „Visual Studio 2022” a fost folosit pentru programarea WebApp-ului și a WebAPI-ului
* „Postman” a fost folosit pentru testarea metodelor implementate de WebAPI.
* „GitHub” a fost folosit pentru source control.
* „SQL Server Management Studio” a fost folosit pentru conectarea la serverul de baze de date și verificarea datelor introduse de WebAPI.
* „ArduinoIDE” a fost folosit pentru programarea microcontrollerului ESP32.
* „Fusion360” a fost folosit pentru realizarea carcasei și a schematicii.

## Prezentare software

* Visual Studio 2022 este un mediu de dezvoltare integrat (IDE) care este foarte popular pentru sarcinile de lucru .NET și C++. Acesta oferă unelte valoroase pentru crearea de cod, precum:
  + Depanarea (debugging) – instrumente puternice precum Live Debugging, Watch, Breakpoints avansați și IntelliTrace ajută la identificarea și rezolvarea erorilor.
  + Completarea codului – IntelliSense și IntelliCode oferă sugestii inteligente, personalizate în funcție de stilul de programare și restul codului.
* Postman este un instrument foarte popular pentru sarcinile de lucru legate de testarea și dezvoltarea API-urilor. Postman oferă o interfață prietenoasă pentru a construi, trimite și vizualiza cereri HTTP precum GET, POST, PUT sau DELETE.
* GitHub este o platformă de găzduire a codului sursă, foarte populară pentru sarcinile de lucru legate de controlul versiunilor și colaborarea între dezvoltatori.
* SQL Server Management Studio (SSMS) este foarte popular pentru sarcinile de lucru legate de gestionarea și dezvoltarea bazelor de date SQL Server. SSMS oferă o interfață grafică intuitivă care permite explorarea și administrarea tabelelor și a altor obiecte dintr-o bază de date SQL Server.
* ArduinoIDE este un mediu de dezvoltare integrat (IDE), foarte popular pentru sarcinile de lucru legate de programarea microcontrolerelor și dezvoltarea proiectelor cu Arduino.
* Fusion360 este un software CAM, CAD și CAE, foarte popular pentru sarcinile de lucru legate de modelarea 3D, proiectarea mecanică, simulare și fabricație.

# Prezentarea aplicației

## WebApp

WebApp-ul reprezintă interfața grafică (front-end-ul) și codul legat de aceasta.

La accesarea inițială, utilizatorul va fi întâmpinat cu pagina de „Welcome”, unde pot fi afișate informații generale.

Acesta are posibilitatea să se logheze pentru a avea acces la mai multe informații, în funcție de rolul pe care îl are (Admin sau Client).

La apăsarea butonului de „Login”, utilizatorul va fi redirecționat către această pagină, unde își va introduce credențialele dacă are un cont deja existent.

După ce utilizatorul s-a logat, acestuia i se va arăta unul din cele două layout-uri, în funcție de rolul său, putând fi client sau admin.

În interfața admin-ului se observă că vor exista mai multe opțiuni, destinate controlului manual al datelor din baza de date pentru toți utilizatorii, permițând modificarea oricăror date.

În interfața clientului, acesta are acces la prezența sa și la cursurile la care este înscris, putând doar vizualiza aceste informații, fără a le putea modifica. El își poate modifica doar parola din pagina cu profilul.

.

Așa arată pagina cu profilul utilizatorului, conținând username-ul, prenumele, numele, instituția, email-ul și numărul de telefon.

Utilizatorul cu rol de client nu va avea acces la link-ul de „Edit”.

Utilizatorul își poate schimba parola prin apăsarea link-ului „Change Password”. Acesta va fi nevoit să-și introducă parola actuală și de două ori parola nouă.

## WebApi

WebAPI-ul gestionează partea de back-end a proiectului, permițând accesul la baza de date. Butoanele de pe WebApp, cât și unele pagini, apelează metode din WebAPI pentru a primi informații din baza de date.  
Metodele din WebAPI sunt de tip HTTP, astfel trebuie selectat ce tip să fie folosit. În realizarea proiectului am folosit metodele: GET (obținere de informații), POST (creare intrări noi în baza de date), PUT (modificarea intrărilor din baza de date) și DELETE (ștergere din baza de date).  
În structura acestuia sunt definiți 4 „utilizatori”, respectiv cei umani: clientul și admin-ul, respectiv scanner-ul, adică microcontroller-ul ESP32, și unul comun.

* „AdminController” este responsabil pentru metodele folosite de WebApp când este logat utilizatorul cu rol de admin.
* „ClientController” este responsabil pentru metodele folosite de WebApp când este logat utilizatorul cu rol de client.
* „CommonController” este responsabil pentru metodele folosite de WebApp atunci când nu este încă logat utilizatorul și conține metode precum resetarea parolei sau logarea.
* „ESP32Controller” conține o singură metodă folosită de microcontroller pentru a introduce în tabelul „Entries” prezența clienților.

### Apelarea funcțiilor de inserare

La apăsarea unui buton sau încărcarea unei pagini, WebApp-ul trimite un HTTP request către WebAPI. Acest request îi este asociat unei funcții care va apela metodele pentru inserarea sau extragerea datelor din baza de date.

.

„Route” definește URL-ul care îi este asociat funcției. De exemplu, funcția „GetAllUsers” are URL-ul „https://localhost:7172/Admin/clients” și este unul de tip GET.

### Inserarea de date

Pentru a introduce și extrage din baza de date, am creat câte o clasă diferită pentru fiecare tabel, în care am injectat „AttendanceContext” pentru a avea acces la baza de date și am definit metodele pe care le vor folosi controlerele.

Clasa moștenește o interfață. Aceasta definește ce funcții trebuie să existe și ce parametri trebuie să aibă.

.

### Baza de date

În crearea și utilizarea bazei de date, am optat pentru „code first approach”, adică structura bazei de date este definită în totalitate de cod.

În prima parte sunt definite tabelele, structura acestora conținând, pe post de coloane, variabilele din interiorul claselor de date.

În a doua parte se stabilesc cheile primare și ce câmpuri să fie unice. De asemenea, de aici se pot adăuga date inițiale, care vor fi introduse o dată cu crearea tabelelor.

### Securitate

Parola utilizatorilor nu trebuie salvată în formă simplă, deoarece reprezintă un risc major în cazul unei spargeri a bazei de date. Din acest motiv, parola trebuie criptată înainte de a fi stocată. Parolei simple i se adaugă un „salt” care poate să fie, de exemplu, data la care se criptează, pentru a face și mai dificilă spargerea, iar apoi este trimisă unui algoritm special care returnează parola criptată.

Din parola „testpass1”, aceasta devine „AQAAAAIAAYagAAAAEKedSrhwVrJx/bhw7PqwY3V9zfw7pR8/5SbMqDnRCW+hq9Vrs7/xybimErcAe9u+Zw==”. Dacă recriptăm din nou parola normală, va fi returnată o altă parolă criptată.

Pentru compararea parolelor (de exemplu, pentru logare), „AspNetCore.Identity”, care este responsabil pentru criptarea parolei, are o funcție de comparare care returnează >0 dacă parolele sunt identice.

În fiecare metodă WebApi se compară un UID care vine împreună cu requestul. Acesta are scopul de a face imposibilă inserarea sau ștergerea de date din baza de date de către cineva/ceva neautorizat. Acesta poate fi asociat cu o parolă necesară la fiecare comunicare.

## Hardware-ul

Hardware-ul constă dintr-un microcontroler ESP32, un ecran LCD 16x2 și un cititor RFID RC522. Microcontrolerul comunică cu LCD-ul prin intermediul protocolului I2C, iar cu cititorul RFID prin protocolul SPI.

Carcasa și schema au fost realizate în Fusion360.

Am realizat coduri diferite pentru scanator: unul dedicat pentru scanarea și înregistrarea prezenței clientului și unul dedicat inscripționării cardurilor.

Scanator admin:

* Scanatorul adminului ia ID-ul cardului care va fi inscripționat prin intermediul serial monitor-ului.
* Nu se folosește ecranul LCD, deoarece orice informație poate fi transmisă prin serial monitor.

Scanator client:

* Scanatorul clientului obține ID-ul cardului prin citirea acestuia de pe card.
* Se apelează WebAPI-ul pentru a înregistra prezența clientului.
* Se confirmă pe ecranul LCD statusul înregistrării.

# Concluzii

Proiectul prezintă o unealtă utilă în cadrul organizațiilor sau instituțiilor, oferind o metodă de supraveghere a activităților.

Interfața grafică face ca aplicația să fie ușor de utilizat, iar utilizarea API-ului pentru legătura dintre WebApp și baza de date facilitează modificarea în cazul în care baza de date va fi mutată pe un server dedicat sau pe un server mai performant.

Scannerul oferă o modalitate ușoară și accesibilă pentru înregistrarea prezenței.

Datorită formatului structurat al aplicației, aceasta permite modificarea și implementarea de funcții noi cu ușurință.

De asemenea, proiectul m-a ajutat să îmi îmbunătățesc metodele de programare și cunoștințele de C#, SQL, C++, și HTML.

# Webografie

<https://www.electronicshub.org/write-data-to-rfid-card-using-rc522-rfid>

<https://arduinogetstarted.com/tutorials/arduino-rfid-nfc>

<https://learn.microsoft.com/en-us/dotnet/fundamentals/networking/http/httpclient>

<https://learn.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-9.0#razor-reserved-keywords>

<https://learn.microsoft.com/en-us/aspnet/core/tutorials/razor-pages/?view=aspnetcore-9.0&viewFallbackFrom=aspnetcore-2.1&preserve-view=true>

<https://randomnerdtutorials.com/esp32-http-get-post-arduino/#http-post>

<https://docs.arduino.cc/learn/electronics/lcd-displays/>

<https://esp32io.com/tutorials/esp32-piezo-buzzer>

<https://medium.com/@codewithankitsahu/implement-entity-framework-a-code-first-approach-in-net-8-api-80b06d219373>

<https://medium.com/@codewithankitsahu/net-8-web-api-crud-operations-125bb3083113>

<https://learningprogramming.net/net/asp-net-core-razor-pages/login-form-with-session-in-asp-net-core-razor-pages/>